Tecnológico de Costa Rica

Sede Central

Escuela de Computación

Curso Diseño de Software - IC6821



Primer Proyecto

Kevin Andrés Hernández Rostrán - 2014092195

Jose Manuel Aguilar Quesada - 2014079312

Jasson Josué Moya Álvarez - 2014082335

Entrega Domingo 24 de Abril

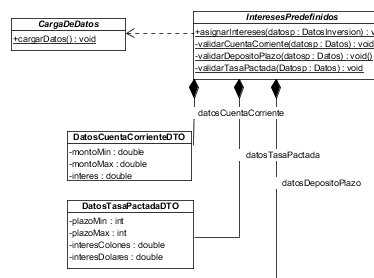
Período Primer Semestre

Prof. Luis Javier Chavarría Sánchez

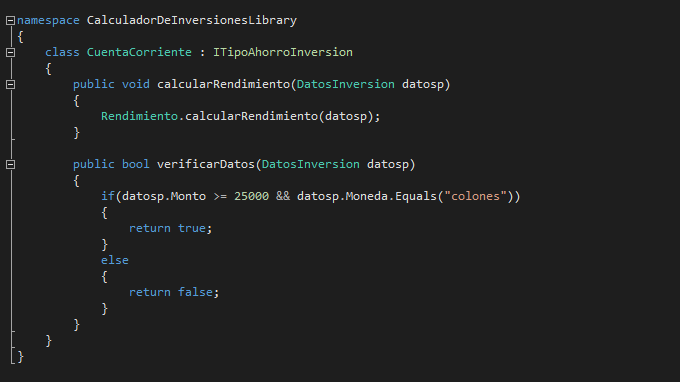
Cumplimiento de S.O.L.I.D.

Single Responsibility Principle

1. Screenshots con partes del diseño UML donde se cumple el principio/patrón.



1. Segmentos de código donde se cumple el principio/patrón.

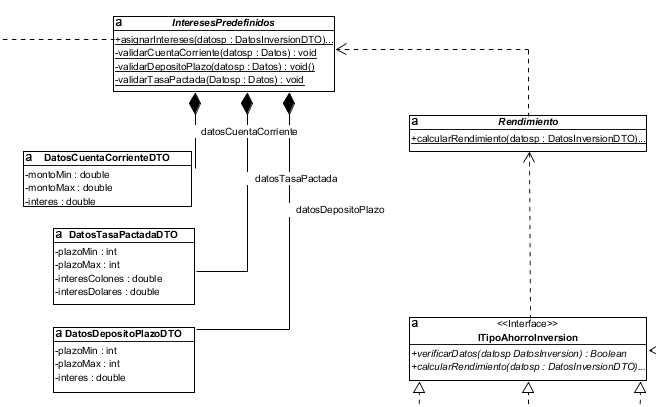


1. Explicación de la estrategias o estrategias tomadas en consideración para el cumplimiento del principio/patrón.

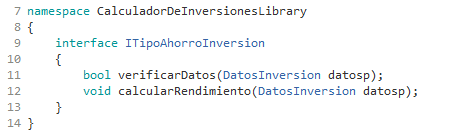
El principio SRP se cumple en todas las clases mediante la creación de métodos especializados con una sola responsabilidad. Por ejemplo, si necesita validar algo, se realiza en un método aparte.

Open/Closed Principle

1. Screenshots con partes del diseño UML donde se cumple el principio/patrón.



1. Segmentos de código donde se cumple el principio/patrón.



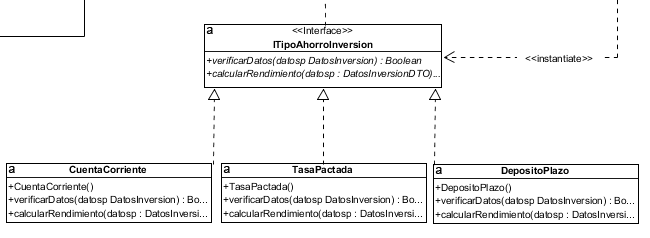


1. Explicación de la estrategias o estrategias tomadas en consideración para el cumplimiento del principio/patrón.

La clase de Rendimiento está abierta a extensión y cerrada a modificación. La clase ITipoAhorroInversion está extendiendo las funcionalidades de Rendimiento.

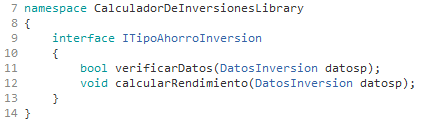
Liskov Substitution Principle

1. Screenshots con partes del diseño UML donde se cumple el principio/patrón.

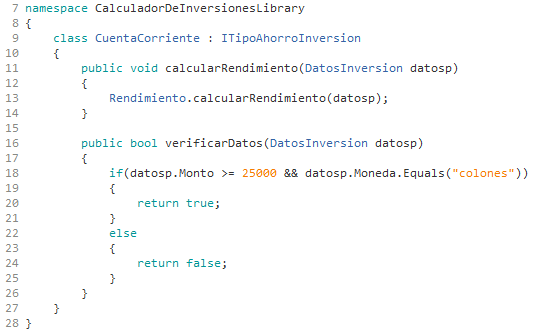


1. Segmentos de código donde se cumple el principio/patrón.

**ITipoAhorroInversion.cs**



**CuentaCorriente.cs**



**TasaPactada.cs**



**DepositoPlazo.cs**



**FabricaSistemasTradicionales.cs**

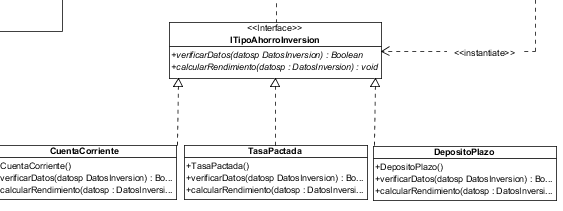


1. Explicación de la estrategias o estrategias tomadas en consideración para el cumplimiento del principio/patrón.

Las clases que utilicen algún tipo de ahorro de inversión como TasaPactada, CuentaCorriente, etc. pueden usar un objeto de tipo ITipoAhoroInversion y utilizar los métodos de sus clases hijas, como se muestra en FabricaSistemasTradicionales.cs

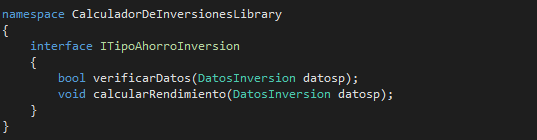
Interface Segregation Principle

1. Screenshots con partes del diseño UML donde se cumple el principio/patrón.



La interface ITipoAhorroInversion tiene solo los métodos necesarios

1. Segmentos de código donde se cumple el principio/patrón.

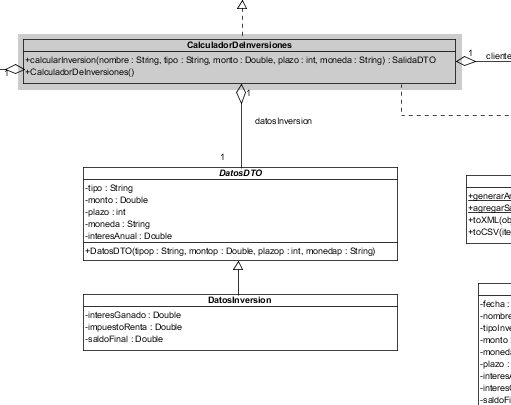


1. Explicación de la estrategias o estrategias tomadas en consideración para el cumplimiento del principio/patrón.

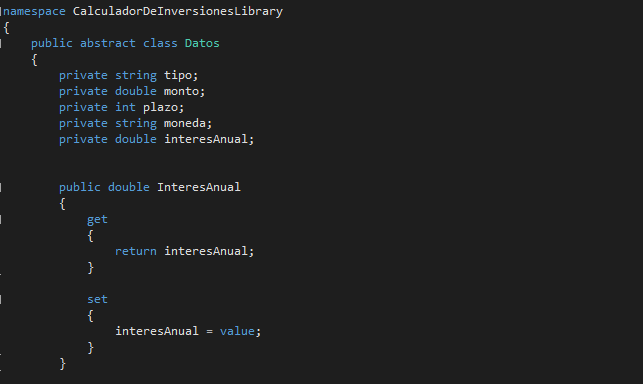
En el proyecto se implementan 3 interfaces, las cuales no tienen más de dos métodos cada una. Esta estrategia se intentó realizar en todas para que cumplieran con el principio.

Dependency Inversion Principle

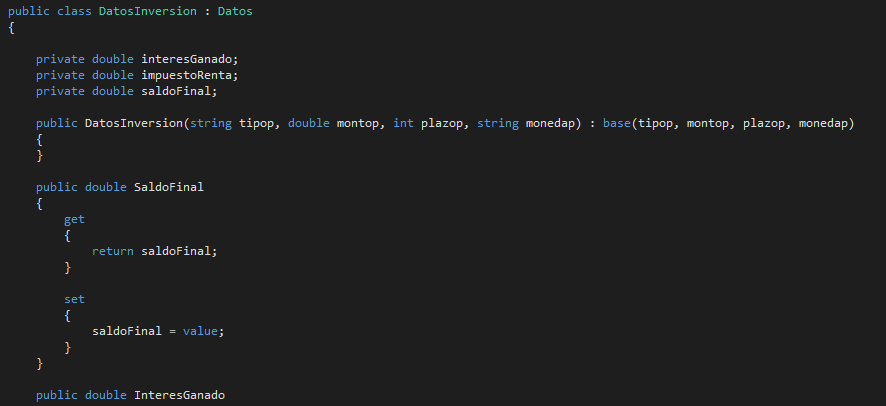
1. Screenshots con partes del diseño UML donde se cumple el principio/patrón.



1. Segmentos de código donde se cumple el principio/patrón.



Clase “Datos” la cual es abstracta y conecta a las clases “DatosInversion” y” CalculadorDeInversiones”



Clase “DatosInversion” que hereda de la clase abstracta “Datos”

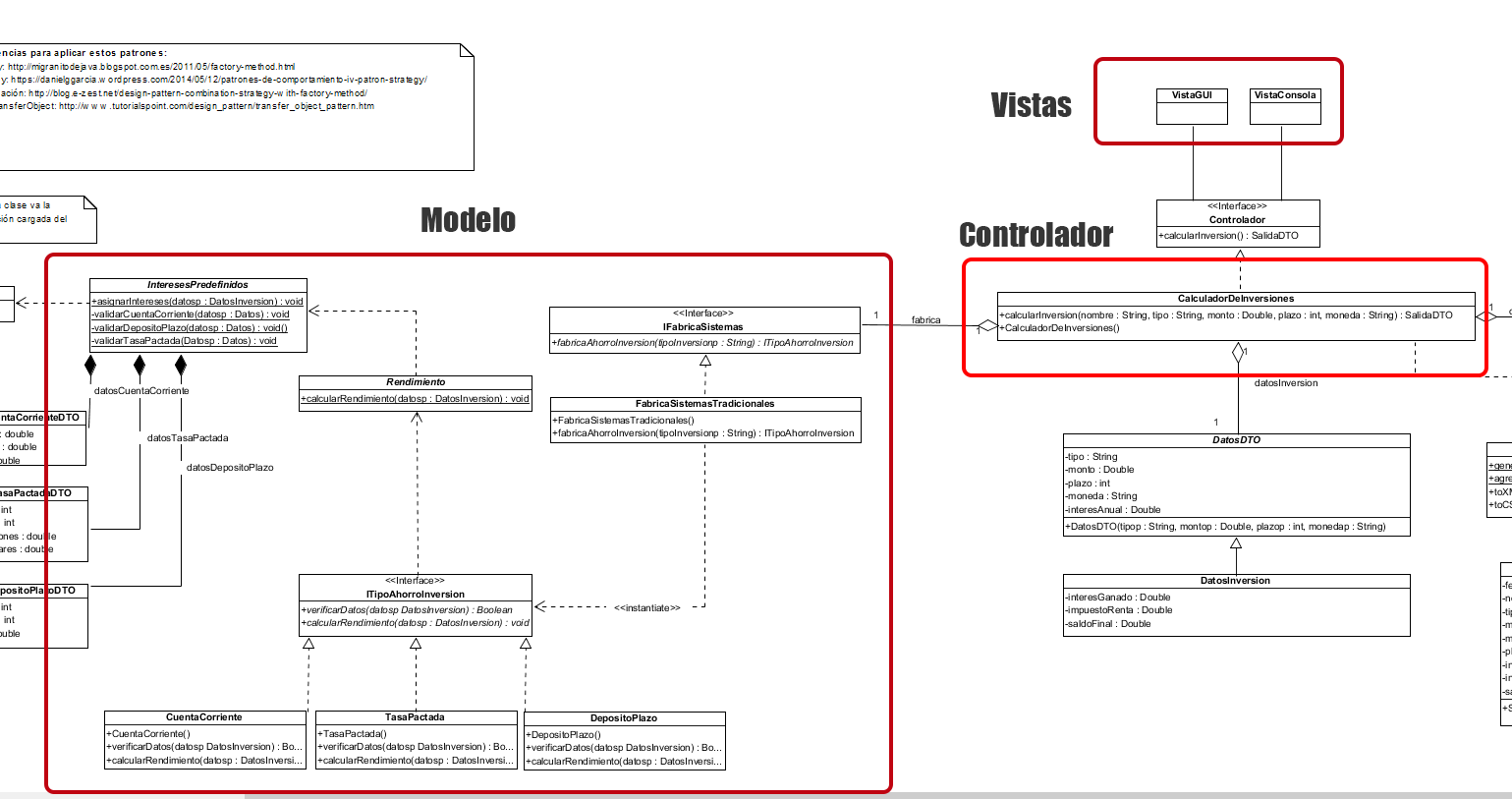
1. Explicación de la estrategias o estrategias tomadas en consideración para el cumplimiento del principio/patrón.

El principio DIP see cumple cuando dos clases concretas se conectan mediante una clase abstracta. Esta fue la estrategia que se siguió durante el diseño del programa. No deben existir clases concretas conectadas entre sí.

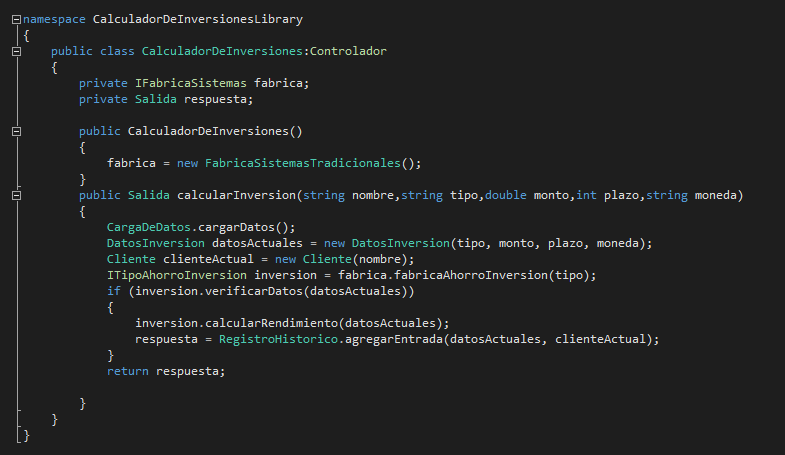
Cumplimiento de Patrones

MVC

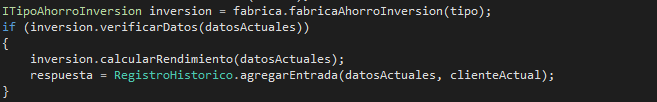
1. Screenshots con partes del diseño UML donde se cumple el principio/patrón.



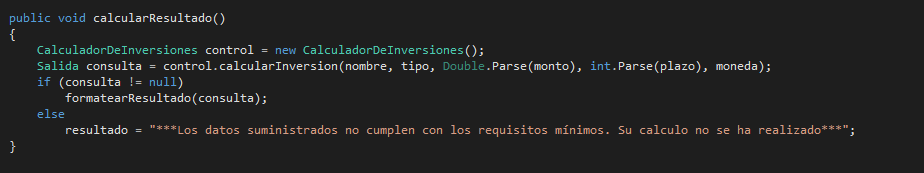
1. Segmentos de código donde se cumple el principio/patrón.



Controlador



Pedir datos del controlador (El cual calcula lo necesario)



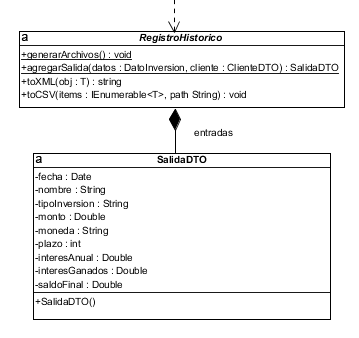
Manda a llamar al controlador desde la vista

1. Explicación de la estrategias o estrategias tomadas en consideración para el cumplimiento del principio/patrón.

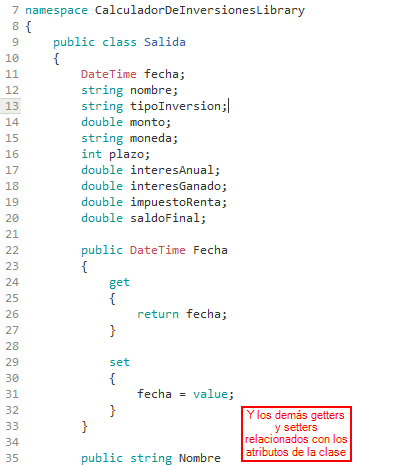
El modelo MVC se implementa mediante el uso de una clase controlador llamada “CalculadorDeInversiones”, la cual llama al modelo, el cual está integrado por distintas clases que implementan sus respectivas acciones. El modelo usa el patrón Strategy y además, Factory. Las vistas de conectan al controlador mediante una instancia nueva.

DTO

1. Screenshots con partes del diseño UML donde se cumple el principio/patrón.



1. Segmentos de código donde se cumple el principio/patrón.

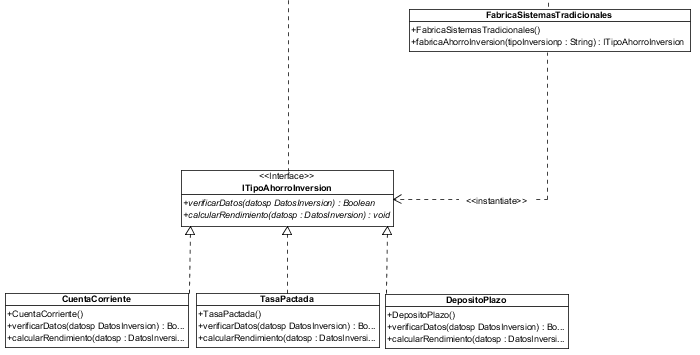


1. Explicación de la estrategias o estrategias tomadas en consideración para el cumplimiento del principio/patrón.

En el ejemplo la clase EntradaDTO es un tipo de dato que puede ser tranferido a través de diferentes clases como sucede en RegistroHistorico.cs

Strategy

1. Screenshots con partes del diseño UML donde se cumple el principio/patrón.

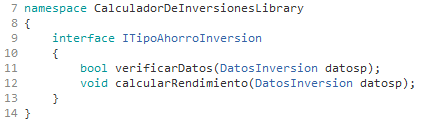


1. Segmentos de código donde se cumple el principio/patrón.

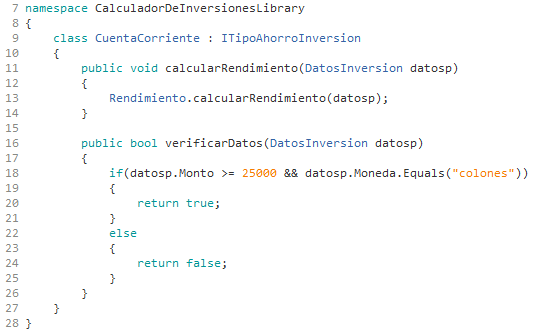
**FabricaSistemasTradicionales.cs**



**ITipoAhorroInversion.cs**



**CuentaCorriente.cs**



**TasaPactada.cs**



**DepositoPlazo.cs**



1. Explicación de la estrategias o estrategias tomadas en consideración para el cumplimiento del principio/patrón.

Como en el proyecto se necesita manejar el comportamiento de una misma clase como diferente (La clase de tipos de inversión), entonces se implementa el patrón Strategy en este aspecto. La interfaz TipoAhorroInversion está implementada por las clases de CuentaCorriente, TasaPactada, DeposivoPlazo. De esta manera, se puede encapsular el comportamiento de dichas clases.